View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Elsevier - Publisher Connector

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 4 (2011) 2196–2205

Procedia Computer Science

International Conference on Computational Science (ICCS 2011)

A New Method for Scheduling Divisible Data on a Heterogeneous Two-Levels Hierarchical System

Amin Shokripour*, Mohamed Othman*,[1], Hamidah Ibrahim, Shamala Subramaniam

Department of Communication Technology and Network,
Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia

Abstract

During the last decade, the use of parallel and distributed systems has become more common. In these systems, a huge chunk of data or computation is distributed among many systems in order to obtain better performance. Dividing data is one of the challenges in this type of systems. Divisible Load Theory (DLT) is proposed method for scheduling data distribution in parallel or distributed systems. In many researches carried out in this field, it was assumed that the used network topology is linear or one-level tree but that is not always true in real systems. In the large scale networks, hierarchical topology is usually used. In this article, we propose a new method which includes a closed-form formula to schedule jobs in a heterogeneous distributed computing system in which its network topology is two-levels hierarchical system. The experiments show the calculated size for each processor is the best.

Key words: Divisible Load Theory, Scheduling, Unreliable Systems, Heterogeneous, Dedicated Systems

1. Introduction

In recent years, there has been considerable specific attention given to solving large computation problems by using distributed computing systems. The researches focused on the use of clusters of workstations, network of workstations and grid environments as candidate systems to do this type of computation. One of the challenges in this field is to find a closed-form formula for distributing data among processors so that the response time (time from admitting job to completing it) is minimum. Methodologies, such as Markovian Queuing Theory, Circuit Theory and Divisible Load Theory (DLT), have been presented for distributing data among processors in parallel and distributed systems so that the response time (time from admitting job to completing it) is minimum.

DLT is based on a specific type of data known as Divisible Load Data. Arbitrarily divisible data are a large amount of data which can be arbitrarily divided into desirable independent parts as each part can be processed separately from the others. Many scientific and engineering applications have this characteristic. Some of these applications

*Corresponding Author
Email addresses: shokripour@gmail.com (Amin Shokripour), mothman@fsktm.upm.edu.my (Mohamed Othman)
[1]The author is also an associate researcher at the Lab of Computational Science and Informatics, Institute of Mathematical Research (INSPEM), Universiti Putra Malaysia.

1877–0509 © 2011 Published by Elsevier Ltd. Open access under CC BY-NC-ND license.
Selection and/or peer-review under responsibility of Prof. Mitsuhisa Sato and Prof. Satoshi Matsuoka
doi:10.1016/j.procs.2011.04.240

are massive experimental data processing, image processing, video processing, mathematical applications, network scheduling, and biological applications [1].

Different models were investigated in DLT researches [2, 1], each of which is made by some assumptions. One installment system with blocking and non-blocking mode communication [3], multi-installment system by non-blocking communication modes [4], system with different processor available time (SDPAT) [5], non-dedicated systems [6], and others are some examples of the investigated models. One of the common network topologies in large scale networks (such as Internet) is the tree network topology. To use this type of network in parallel and distributed computing systems, we should have a method for scheduling jobs.

In this research, we present a scheduling method to schedule jobs in a heterogeneous multi-installment system with the tree network topology. The proposed method consider four necessary steps for scheduling multi-installment systems. It attempts to find the proper number of processors, the proper number of installments, the size of the assigned task for internal installments, and the size of the assigned task for last installment. Finally, we show that the proposed formula is true.

The remainder of this paper is organized as detailed below. Section 2 presents related works. Our model and notations are introduced in the third section. In section 4 the proposed method for two-level hierarchical systems, which includes a formula for calculating the proper size of task for each branch and processors ordering algorithm, is presented. The results of experiments and their analysis appear in section 5. In the last section, conclusion is obtainable.

2. Related Works

Hierarchical network topology is one of the commonly used topologies in large scale networks. It was investigated from the first years of DLT [7]. One of the applications of tree topology is wireless networks. These types of networks were investigated by Katsaros and Polyzos [8].

Some restrictions were defined on this topology and were converted it to some other customized topologies and each of the new topologies was separately investigated for scheduling by using DLT. The simplest resulted topology from the assumptions is the one-level tree. A one-level tree includes a root and some leaves, each of which is directly connected to the root. This topology is one of the commonly investigated topologies in researches about using DLT for scheduling parallel and distributed systems [3].

Another investigated hierarchical topology is the spanning tree. A spanning tree is produced by defining a condition on a complex network topology. As defined in graph theory, it is a network in which each of the clients is only connected to one other worker and the connection is made by using the middle nodes. The first attempt in this field was presented by Yao et al. [9]. Yao et al. tried to do job scheduling in an arbitrary network. For this reason, they made a spanning tree from the network and scheduled the jobs in the spanning tree network. After Yao's research, England et al. presented another study about spanning trees [10]. In the newer research, two different spanning trees (Traditional Spanning Tree and Robust Spanning Tree) were investigated by England et al. and two different algorithms (Centralized and Distributed) were proposed to schedule these types of trees. They compared their method with Yao's method and showed the response time for their method is better than Yao's method. The last research about job scheduling using spanning trees was published by Viswanathan et al [11]. Resource-aware optimal load distribution strategy based on optimal sequencing (RAOLD-OS), which has been shown to provide optimal processing time solutions for a given spanning tree network [9], was applied to some different spanning tree routing strategies such as Minimum Spanning Tree (MST), Shortest Path spanning Tree (SPT), Fewest Hops spanning Tree (FHT), and Robust Spanning Tree (RST). It was shown that the SPT routing strategy has the better trade-off between complexity and performance while each of the other strategies has better trade-off between other parameters.

Tree network topology is one of the complex network topologies. Job scheduling in this topology is so difficult because of its complexity and scheduling complexity. One of the ten reasons for using Divisible Load Theory is named Equivalent Networks [2]. This means that we can replace a branch of a tree with an equivalent node so that it works the same as all the member of branch. It was based on some researches. Suresh et al. presented a method which used this characteristic of DLT [12]. Jia et al. investigated a multi-level tree in which resources were unknown [13]. They defined two types of parameters for the network: static and dynamic. An equivalent tree was made to show the available resources and the scheduling method was applied on this tree.

Table 1: Notations

Notation	Description
W	Total size of data
W_i	Total size of data for the P_i^s
V_i	Size of each installment for the P_i^s
n_i	Number of installments for the P_i^s
m_i	Number of processors for the P_i^s
l	Number of branches
α_i^j	It is the size of allocated fraction to processor P_i in each internal installment in the group j.
β_i^j	It is the size of allocated fraction to processor P_i in the last installment in the group j.
w_i^j	Ratio of the time taken by processor P_i in the group j, to compute a given load, to the time taken by a standard processor, to compute the same load
z_i^j	Ratio of the time taken by P_i in the group j, to communicate a given load, to the time taken by a standard link, to communicate the same load
s_i^j	Computation overhead for processor P_i in the group j
o_i^j	Communication overhead for processor P_i in the group j
$T(W_i)$	This is the response time for a task with size W_i.

3. Preliminaries

3.1. Notations

In this paper, several notations and their definitions are described in Table 1.

3.2. Model

In this research, the two-level tree network topology is investigated. For the two-level tree network topology, we use a model the same as Fig. 1.

The used system is a communication based heterogeneous multi-installment which includes communication and computation overheads and the communication mode is blocking.

3.3. A Review on the Formulation of Multi-Installment Systems

In one of the our previous research, we reached a closed-form formula for scheduling jobs in a heterogeneous system which includes over heads and communication mode is blocking [14]. In this section we will review them because in the next sections we need them.

For scheduling internal installments these formula can be used:

$$\begin{cases} \Delta_i = \frac{z_1+w_1}{z_i+w_i} \\ \Phi_i = \frac{o_1+s_1-o_i-s_i}{z_i+w_i} \\ \alpha_1 = \frac{1-\frac{1}{V}\sum_{i=2}^{m}\Phi_i}{1+\sum_{i=2}^{m}\Delta_i} \\ \alpha_i = \alpha_1\Delta_i + \frac{1}{V}\Phi_i \end{cases} \tag{1}$$

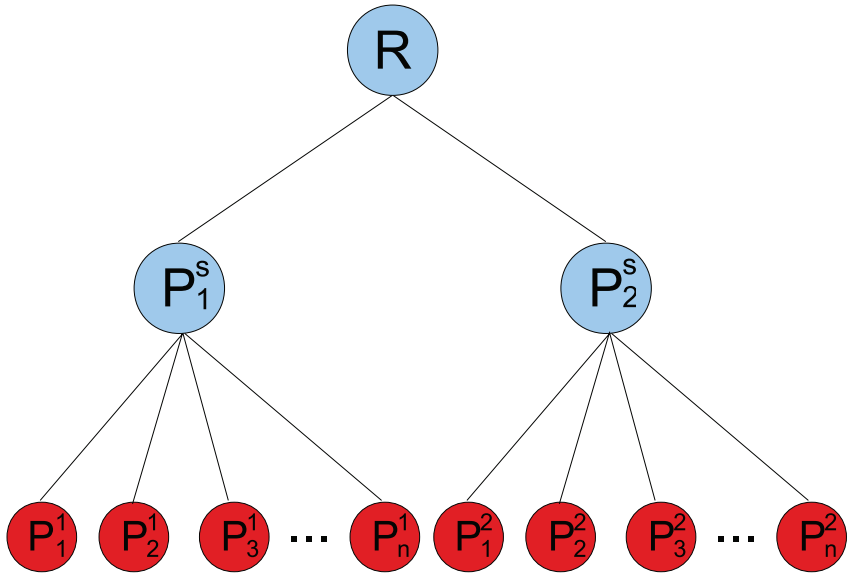

Figure 1: Two-Level Hierarchical Network Topology

Last installment can be scheduled by using these formulas:

$$\begin{cases} \delta_{i+1} = \frac{s_i-(s_{i+1}+o_{i+1})}{w_{i+1}+z_{i+1}} \\ \varepsilon_{i+1} = \frac{w_i}{w_{i+1}+z_{i+1}} \\ \mathrm{E}_i = \prod_{j=2}^{i} \varepsilon_j \\ \Gamma_i = \sum_{j=2}^{i} (\delta_j \prod_{k=j+1}^{i} \varepsilon_k) \\ \beta_i = \mathrm{E}_i\beta_1 + \frac{1}{V}\Gamma_i, i = 2, \ldots, m \\ \beta_1 = \frac{1-\frac{1}{V}\sum_{i=2}^{m}\Gamma_i}{1+\sum_{i=2}^{m}\mathrm{E}_i} \end{cases} \tag{2}$$

In the above formulas, m is the proper number of processors. We use this formula for calculating its value.

$$W \geq \left(\frac{(1+\sum_{i=2}^{m}\Delta_i)(\sum_{j=2}^{m}(\Phi_j z_j + o_j) - s_1)}{(w_1 - \sum_{j=2}^{m}\Delta_j z_j)} + \sum_{i=2}^{m}\Phi_i\right)^2. \\ \frac{(z_1+w_1)(\sum_{i=2}^{m-1}\mathrm{E}_i - \sum_{i=2}^{m}\Delta_i)}{(1+\sum_{i=2}^{m-1}\mathrm{E}_i)[(z_1+w_1)\sum_{i=2}^{m}\Phi_i - (1+\sum_{i=2}^{m}\Delta_i)(o_1+s_1)]} \tag{3}$$

We started from $m = 1$ and calculate non-equation. If for the $m = M$ the non-equation is not true, we found that $M - 1$ is the proper number of the processors.

The proper number of installment is another important parameter for jobs scheduling in a multi-installment system. The used formula for calculating this parameter is the same as below

$$\frac{W(z_1+w_1)(\sum_{i=2}^{m-1}\mathrm{E}_i - \sum_{i=2}^{m}\Delta_i)}{(1+\sum_{i=2}^{m-1}\mathrm{E}_i)[(z_1+w_1)\sum_{i=2}^{m}\Phi_i - (1+\sum_{i=2}^{m}\Delta_i)(o_1+s_1)]} = (n+1)^2 \tag{4}$$

3.4. Response Time Function for a One-Level Tree Topology

DLT is based on this concept that for the best response time all the processors should finish their tasks at the same instant time. Hence, we need to know the response time in one-level tree topology. The response time for the multi-installment systems includes required time for internal installments and last installment. By using Eq.(1) and

Eq.(2), we can write the function of response time for a multi-installment system with one-level tree network topology same as Eq.(5).

$$T(W_1) = n_1\left(\sum_{j=2}^{m_1}(\alpha_j^1 V_1 z_j^1 + o_j^1) + \alpha_1^1 V_1 z_1^1 + o_1^1\right) + \left(\frac{V_1 - \sum_{i=2}^{m_1}\Gamma_i^1}{1 + \sum_{i=2}^{m_1}\mathrm{E}_i^1}\right)(z_1^1 + w_1^1) + (o_1^1 + s_1^1) \tag{5}$$

Because one of the unknown variables in this research is the size of assigned data to each group, we change this formula to a formula based on W_1. We define a new variable the same as below

$$\tau_1 = \sqrt{\frac{1}{(1 + \sum_{i=2}^{m_1}\mathrm{E}_i)}}\cdot \sqrt{\frac{(1 + \sum_{i=2}^{m_1}\Delta_i)(z_1 + w_1) - (1 + \sum_{i=2}^{m_1}\mathrm{E}_i)\sum_{j=1}^{m_1}(\Delta_j z_j)}{(1 + \sum_{i=2}^{m_1}\Delta_i^1)(\sum_{j=2}^{m_1}(\Phi_j^1 z_j^1) + \sum_{j=1}^{m_1} o_j^1) - \sum_{i=2}^{m_1}\Phi_i^1 \sum_{j=1}^{m_1}(\Delta_j^1 z_j^1)}} \tag{6}$$

From Eq.(6) and Eq.(4), we find that

$$n + 1 = \sqrt{W_1}\tau_1 \tag{7}$$

After rewriting Eq.(5), we have

$$\begin{aligned} & W_1\frac{\sum_{j=1}^{m_1}(\Delta_j^1 z_j^1)}{(1 + \sum_{i=2}^{m_1}\Delta_i^1)} + \sqrt{W_1}2\tau_1 \\ & \left(\frac{(1 + \sum_{i=2}^{m_1}\Delta_i^1)(\sum_{j=2}^{m_1}(\Phi_j^1 z_j^1) + \sum_{j=1}^{m_1} o_j^1) - \sum_{i=2}^{m_1}\Phi_i^1 \sum_{j=1}^{m_1}(\Delta_j^1 z_j^1)}{1 + \sum_{i=2}^{m_1}\Delta_i^1}\right) \\ & - \frac{(1 + \sum_{i=2}^{m_1}\Delta_i^1)(\sum_{j=2}^{m_1}(\Phi_j^1 z_j^1) + \sum_{j=1}^{m_1} o_j^1) - \sum_{i=2}^{m_1}\Phi_i^1 \sum_{j=1}^{m_1}(\Delta_j^1 z_j^1)}{1 + \sum_{i=2}^{m_1}\Delta_i^1} \\ & - \frac{(z_1^1 + w_1^1)\sum_{i=2}^{m_1}\Gamma_i^1}{1 + \sum_{i=2}^{m}\mathrm{E}_i^1} + (o_1^1 + s_1^1) = T(W_1) \end{aligned} \tag{8}$$

Three new symbols are defined to make Eq.(8) simpler.

$$\begin{cases} F_1 = -\dfrac{(1 + \sum_{i=2}^{m_1}\Delta_i^1)(\sum_{j=2}^{m_1}(\Phi_j^1 z_j^1) + \sum_{j=1}^{m_1} o_j^1)}{1 + \sum_{i=2}^{m_1}\Delta_i^1} \\ \qquad + \dfrac{\sum_{i=2}^{m_1}\Phi_i^1 \sum_{j=1}^{m_1}(\Delta_j^1 z_j^1)}{1 + \sum_{i=2}^{m_1}\Delta_i^1} - \dfrac{(z_1^1 + w_1^1)\sum_{i=2}^{m_1}\Gamma_i^1}{1 + \sum_{i=2}^{m}\mathrm{E}_i^1} + (o_1^1 + s_1^1) \\ G_1 = 2\tau_1\left(\dfrac{(1+\sum_{i=2}^{m_1}\Delta_i^1)(\sum_{j=2}^{m_1}(\Phi_j^1 z_j^1)+\sum_{j=1}^{m_1} o_j^1)-\sum_{i=2}^{m_1}\Phi_i^1 \sum_{j=1}^{m_1}(\Delta_j^1 z_j^1)}{1+\sum_{i=2}^{m_1}\Delta_i^1}\right) \\ H_1 = \dfrac{\sum_{j=1}^{m_1}(\Delta_j^1 z_j^1)}{1+\sum_{i=2}^{m_1}\Delta_i^1} \end{cases} \tag{9}$$

We can rewrite Eq.(8) the same as below

$$H_1 W_1 + \sqrt{W_1}G_1 + F_1 = T(W_1) \tag{10}$$

4. The New Scheduling Method for Hierarchical Systems

4.1. A New Method for Scheduling Two-Level Hierarchical Systems

For scheduling a multi-installment system, we should calculate four parameters; the proper number of installments, the proper number of processors, the size of the assigned task to each processor for internal installments and the size of the assigned task to each processor in last installment.

Before finding these variables, we should sort the processors because it has direct affect on the response time of the systems. Mingsheng showed ordering processor by increasing z_i is the best method for job scheduling in one installment system [3]. Therefore, we sort the processors of each branch by increasing z_i. To find the first parameter, we use the proposed closed-form formula in [14] to calculate the proper number of installments; Eq.(4) for the first branch and select it as the proper number of installments. We use also Eq.(3) for the first branch to answer the second question. In this section, we present a method to answer the third and fourth questions.

As mentioned, in DLT researches, all the processors are forced to finish their tasks at the same instant in time. Therefore, in this research, we attempted to find the size of the assigned data to each branch so that all the branches finish their tasks simultaneously. Response time of each branch can be found by using Eq.(10). The time of participation by each branch in the task is equal to its response time plus the time of communication between the root of the branch and the main root as shown in Eq.(11).

$$\begin{cases} H_1W_1 + G_1\sqrt{W_1} + F_1 = H_2W_2 + G_2\sqrt{W_2} + F_2 + W_2z_2^2 + o_2^2 \\ H_2W_2 + G_2\sqrt{W_2} + F_2 = H_3W_3 + G_3\sqrt{W_3} + F_3 + W_3z_3^3 + o_3^3 \\ \dots \\ H_{l-1}W_{l-1} + G_{l-1}\sqrt{W_{l-1}} + F_{l-1} = H_lW_l + G_l\sqrt{W_l} + F_l + W_lz_l^l + o_l^l \end{cases} \tag{11}$$

By using these equations, we can calculate W_i based on W_1.

$$W_i = Z_iW_1, i = 2, 3, \dots, l-1 \tag{12}$$

We know that

$$W_1 + W_2 + W_3 + \dots + W_l = \sum_{i=1}^{l} W_i = W \tag{13}$$

hence,

$$W_2 = W - (1 + \sum_{i=3}^{l} Z_i)W_1 \tag{14}$$

To make the formulas simple, we define $\Psi = (1 + \sum_{i=3}^{l} Z_i)$.

$$\begin{aligned} H_1W_1 + G_1\sqrt{W_1} + F_1 = \\ H_2(W - \Psi W_1) + G_2\sqrt{W - \Psi W_1} + F_2 \\ + z_2^2(W - \Psi W_1) + \sqrt{W - \Psi W_1} + o_2^2 \end{aligned} \tag{15}$$

We solve this equation to find the value of W_1.

$$\begin{aligned} &[H_1 + (H_l + z_2^2)\Psi]W_1 + G_1\sqrt{W_1} + [F_1 - (H_l + z_2^2)W - F_l - o_2^2] \\ &- G_l\sqrt{W - \Psi W_1} = 0 \end{aligned} \tag{16}$$

We define two new symbols and rewrite Eq.(16).

$$\begin{cases} H = H_1 + (H_l + z_2^2)\Psi \\ F = F_1 - (H_l + z_2^2)W - F_l - o_2^2 \end{cases} \tag{17}$$

$$HW_1 + G_1\sqrt{W_1} + F - G_l\sqrt{W - \Psi W_1} = 0 \tag{18}$$

$$(HW_1 + F)^2 = (G_l\sqrt{W - \Psi W_1} - G_1\sqrt{W_1})^2 \tag{19}$$

$$\begin{aligned} &-(HW_1)^2+(WG_l^2-F^2)\\ &+(G_1^2-2FH-\Psi G_l^2)W_1=2G_lG_1\sqrt{WW_1-\Psi W_1^2} \end{aligned} \tag{20}$$

Three new symbols are necessary to make the Eq.(20) simpler.

$$\begin{cases} P=WG_l^2-F^2\\ T=G_l^2-2FH-\Psi G_l^2\\ X=G_lG_1 \end{cases} \tag{21}$$

Eq.(20) can be written as below:

$$[P-(HW_1)^2+TW_1]^2=(2X\sqrt{WW_1-\Psi W_1^2})^2 \tag{22}$$

$$\begin{aligned} &H^4W_1^4+(T^2-2PH^2+4X^2\Psi)W_1^2\\ &-2H^2TW_1^3+(2PT-4X^2W)W_1+P^2=0 \end{aligned} \tag{23}$$

After defining five new symbols, we have a simple quartic equation, Eq.(25).

$$\begin{cases} A=H^4\\ B=-2H^2T\\ C=T^2-2PH^2+4X^2\Psi\\ D=2PT-4X^2W\\ E=P^2 \end{cases} \tag{24}$$

$$AW_1^4+BW_1^3+CW_1^2+DW_1+E=0 \tag{25}$$

Eq.(25) is a quartic equation that we can use to find its roots using Ferrari's method. The root of this equation, which is larger than zero and if Eq.(13) holds true, is acceptable. By using the root of Eq.(25) in Eq.(12), we can find the value of $W_i, i=2,3,\ldots,l$. This means that we have the size of data for each group, and by using Eq.(1) and Eq.(2) , the job can be scheduled. This method is called the *the Quartic Method* because quartic equation is used.

5. Results and Discussion

For the experiments, we used a simulator implemented by C++ which runs with Linux-based operating system, Suse 11.1. A set of 50 processors with attributes which were produced randomly, was used as input for simulator. A set of jobs was used as the second input. Value of w is ten times as large as z value for all processors. This means that communication speed is much faster than computation speed. We did all the experiments in a two-level hierarchical system. Members of each branch and the roots were randomly selected from the pool of workers.

We attempted to show the Quartic Method works correctly and has a response time that is the shortest. As mentioned, in scheduling multi-installment systems four questions should be answered. In this section therefore, we checked whether the Quartic Method offers the best value for each of the parameters, the proper number of processors, the proper number of installments, and the proper size of task for the internal and last installment. We did three sets of experiments to show the method gives us the best value for each parameter.

As can be seen in Fig. 2, the calculated value for the size of task for each of the branches while used the Quartic Method (zero point in the graphs) is much better than the other points. In this experiment, we manually changed the size of the assigned task to each branch and the response times for the changed size of tasks were calculated and shown in the graphs. We reduced 5%, 10% and 15% of the assigned task to the first branch and added them to the second branch. We also added these percentages of the size of the assigned tasks to the first branch and reduced the percentages in the second branch.

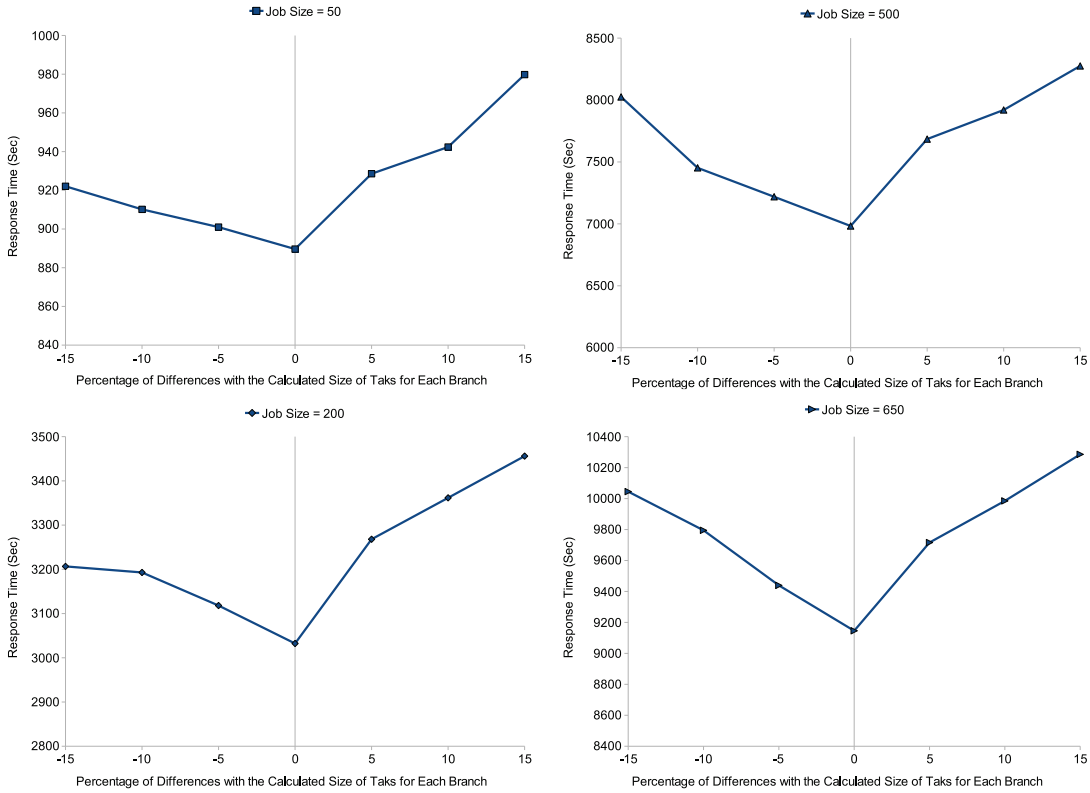

Figure 2: Response Time vs Differences with the Calculated Size of the Assigned Task using the Quartic Method for Different Sizes of Jobs

By using an experiment similar to the above experiment, we can evaluate the Quartic Method's efficiency in calculating the proper number of installments. Fig. 3 shows that due to the same reason as was mentioned for the calculated size of task for each processor, it is clear that the Quartic Method presents the best proper number of installments for a heterogeneous multi-installment system with a two-level hierarchical network topology.

The Quartic Method can be evaluated for its ability to find the proper number of processors with the similar experiment. The results of this experiment can be seen in Fig. 4. As shown in the graphs, we manually added one, two, three and four processors to the calculated proper number of processors and also reduced the same number of processors. The zero points show the response times for the proper numbers of processors calculated using the Quartic Method.

6. Conclusion

One of the used topologies in mobile systems is hierarchical. Job scheduling in this topology (in which the system includes overhead, communication mode is blocking, and root uses multi-installment method to send the tasks) was not studied before this. In this study, we presented a new methods for job scheduling in two-level tree network topology. The presented method uses a quartic close-form formula to find the proper size of task for each branch. Then by using Eq.(1) and Eq.(2), each branch is independently scheduled. Our experiments showed that the proposed method present the best scheduling.

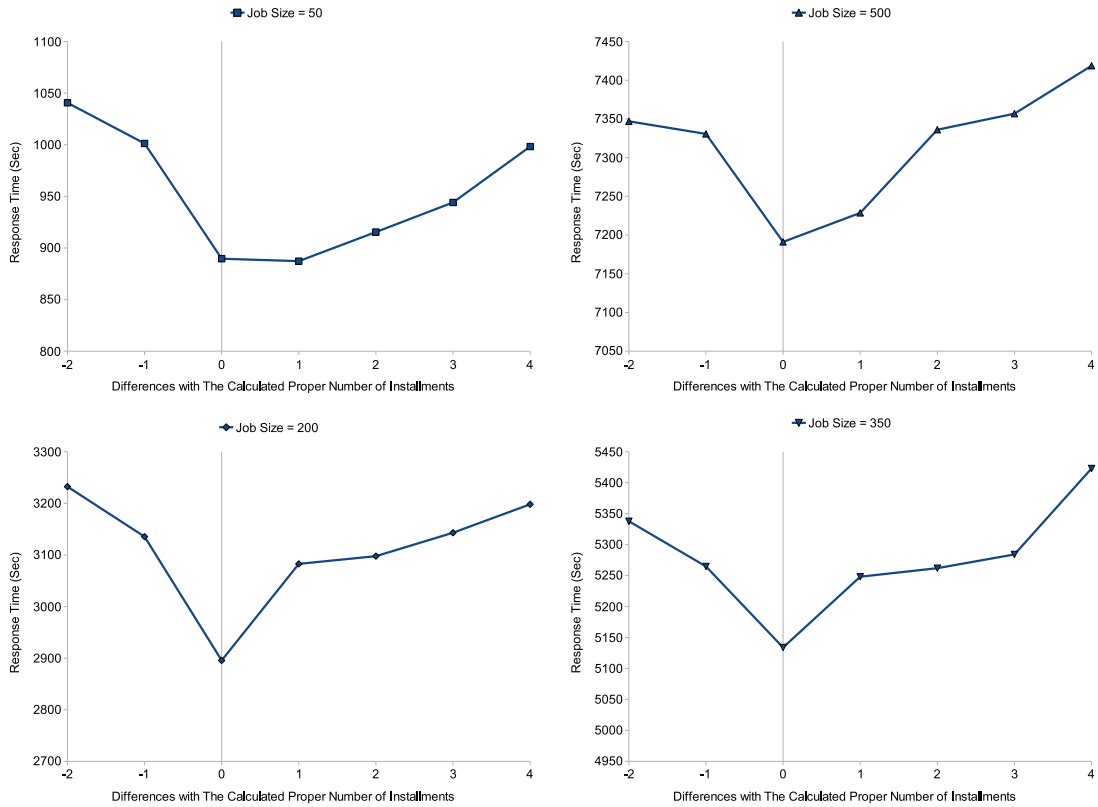

Figure 3: Response Time vs Differences with the Calculated Proper Number of Installments using the Quartic Method for Different Sizes of Jobs

Acknowledgment

The research was supported by the Malaysian Ministry of Higher Education, Fundamental Research Grant (FRGS) 01-11-09-734FR

References

[1] A. Shokripour, M. Othman, Categorizing DLT researches and its applications, European Journal of Scientific Research 37 (3) (2009) 496–515.
[2] T. Robertazzi, Ten reasons to use divisible load theory, COMPUTER 36 (5) (2003) 63–68.
[3] S. Mingsheng, Optimal algorithm for scheduling large divisible workload on heterogeneous system, APPL MATH MODEL 32 (2008) 1682–1695.
[4] S. Mingsheng, S. Shixin, Optimal multi-installments algorithm for divisible load scheduling, in: Eighth International Conference on High-Performance Computing in Asia-Pacific Region, 2005, pp. 45–54.
[5] A. Shokripour, M. Othman, H. Ibrahim, A new algorithm for divisible load scheduling with different processor available times, in: Lecture Notes in Computer Science, Vol. 5990, Springer, Berlin / Heidelberg, 2010, pp. 221–230.
[6] A. Shokripour, M. Othman, H. Ibrahim, S. Subramaniam, A new method for job scheduling in a non-dedicated heterogeneous system, in: Procedia Computer Science, Vol. 3, Elsevier B.V., 2011, pp. 271–275.
[7] S. Bataineh, T. Hsiung, T. Robertazzi, Closed form solutions for bus and tree networks of processors load sharing a divisible job, IEEE Trans. Computers. 43 (10) (1994) 120–131.
[8] K. Katsaros, G. C. Polyzos, Optimizing operation of a hierarchical campus-wide mobile grid, in: 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, Greece, 2007.
[9] J. Yao, B. Veeravalli, Design and performance analysis of divisible load scheduling strategies on arbitrary graphs, CLUSTER COMPUT 7 (2) (2004) 191–207.
[10] D. England, V. Bharadwaj, J. B. Weissman, A robust spanning tree topology for data collection and dissemination in distributed environments, IEEE T PARALL DISTR 18 (5) (2007) 608–620.

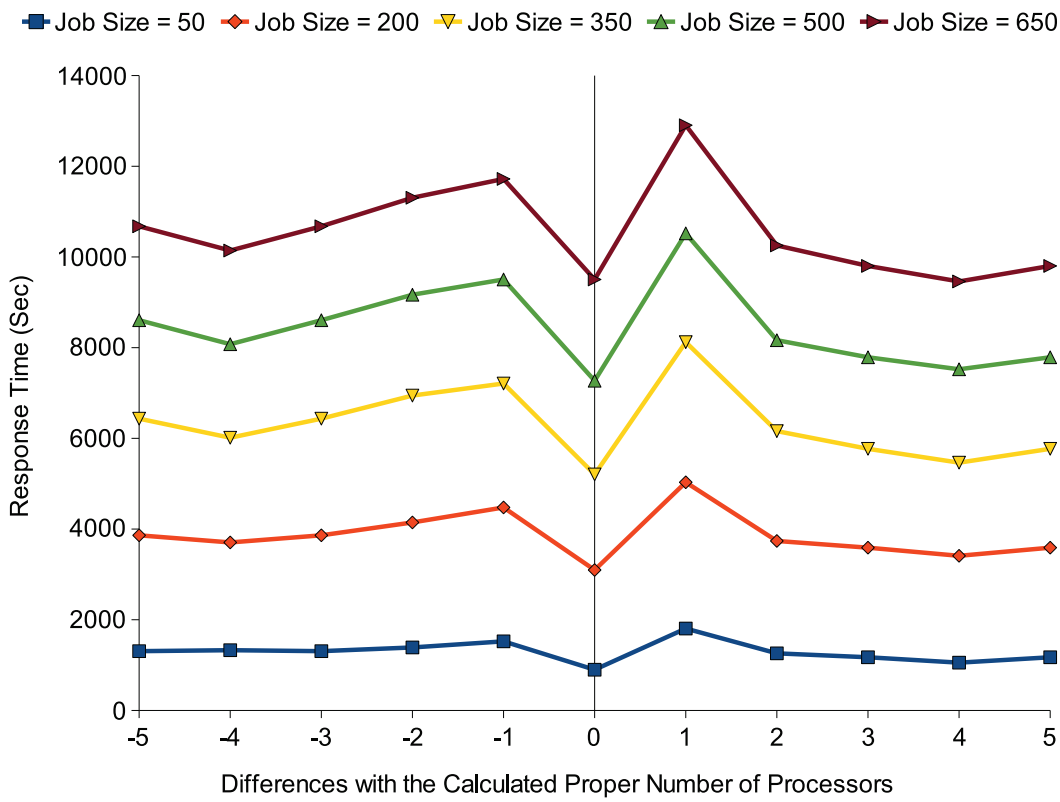

Figure 4: Response Time vs Differences with the Calculated Proper Number of Processors using the Quartic Method for Different Sizes of Jobs

[11] S. Viswanathan, B. Veeravalli, J. Jingxi, Spanning tree routing strategies for divisible load scheduling on arbitrary graphs, a comparative performance analysis, in: International Conference on High Performance Computing (HiPC), 2009, pp. 50 –58. doi:10.1109/HIPC.2009.5433225.

[12] S. Suresh, V. Mani, S. N. Omkar, H. J. Kim, An equivalent network for divisible load scheduling in nonblocking mode of communication, Computers and Mathematics With Applications 49 (10) (2005) 1421–1431.

[13] J. Jia, B. Veeravalli, D. Ghose, Adaptive load distribution strategies for divisible load processing on resource unaware multilevel tree networks, IEEE T COMPUT 56 (7) (2007) 999–1005.

[14] A. Shokripour, M. Othman, H. Ibrahim, S. Subramaniam, A method for scheduling heterogeneous multi-installment systems, in: Lecture Notes In Artificial Intelligence (LNAI), Vol. 6592, Springer, Berlin / Heidelberg, 2011, pp. 31–41.